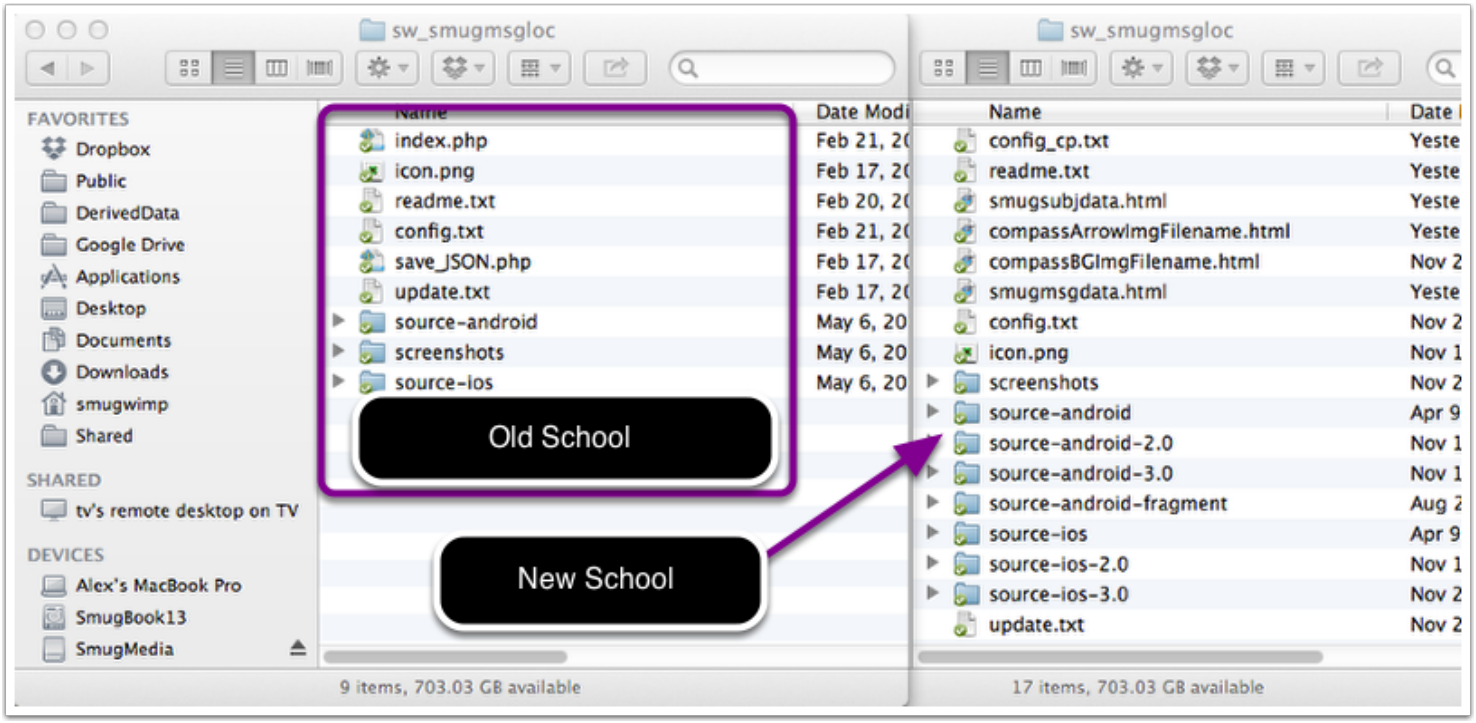
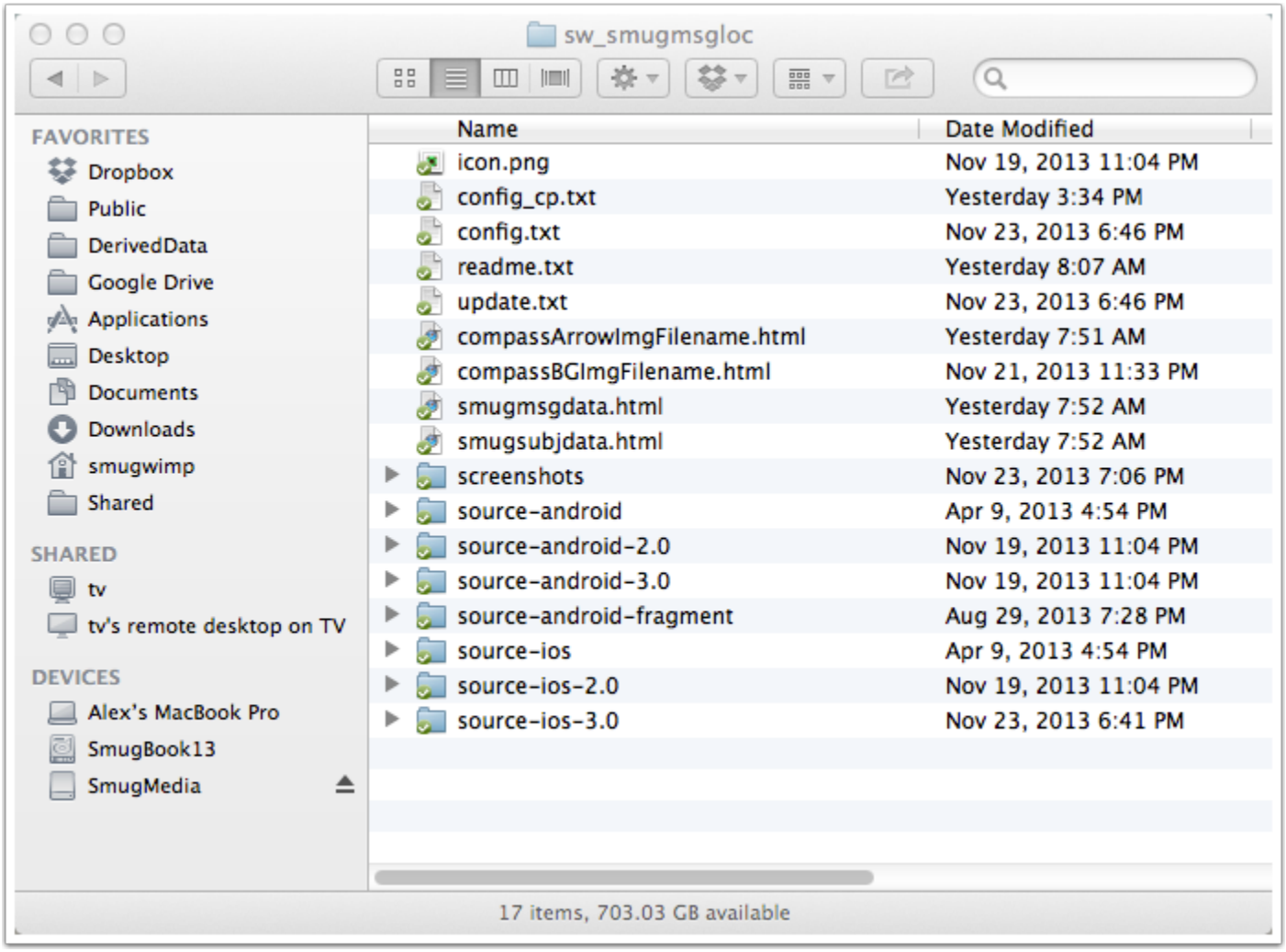
Connecting your Plugin Control Panel to your BT Plugin Code

A Quick Guide on getting User Variables into your project.



With BT v3 comes a slightly different method in connecting your user defined options from the Control Panel into your BT Plugin. Gone is the old index.php method, however the new method is not too different, so you should be able to make the transition smoothly, if you keep a few things in mind. But we'll start from scratch and connect two items for you so that you can get a feel for how it's done.

Anatomy of your plugin directory...



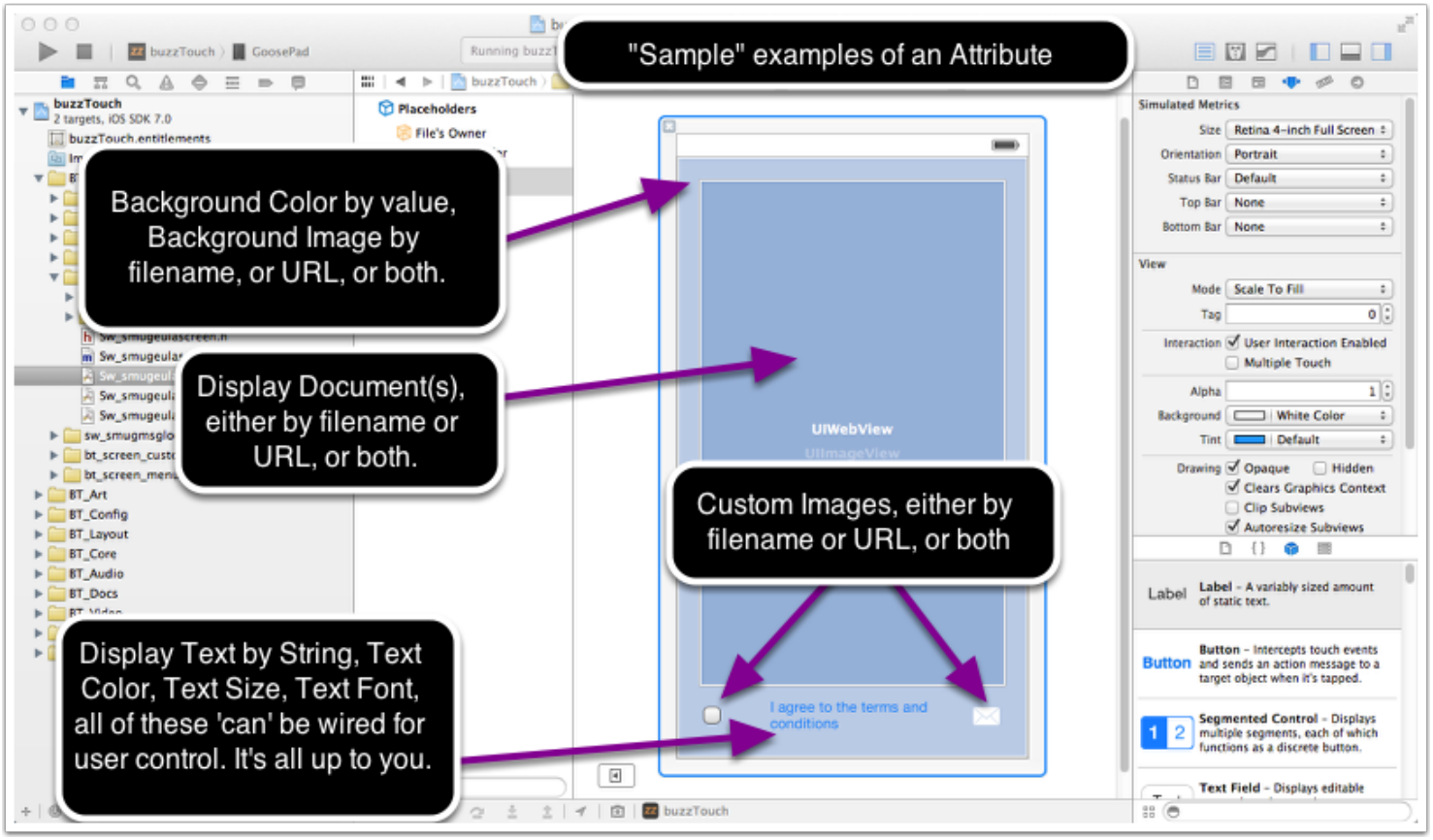
01) icon.png - Same as before
02) config_cp.txt - in a rough sense, the replacement for 'index.php'
03) config.txt - same as before, but with a new attribute: "supportedDevices"
04) readme.txt - same as before
05) update.txt - same as before
06) All HTML files are 'custom sections' of your control panel. To be discussed. It's why we're here.
07) Screenshots directory - same as before
08) source-android - same as before; to be laid out in BT Android v2 format, for backward compatibility with BTv2 projects
09) source-android-2.0 - to be laid out in BT Android v2 format, for backward compatibility with BTv2 projects (activities)
10) source-android-3.0 - to be laid out in BT Android v3 format (Android API 11, using fragments)
11) source-android-fragment - to be laid out in BT Android v3 format (Android API 11, using fragments)
12) source-ios - same as before; to be laid out in BT iOS v2 format

13) source-ios-2.0 - to be laid out in BT iOS v2 format
14) source-ios-3.0 - to be laid out in BT iOS v3 format (which is in essence the same as v2 format, with updated code)

And a quick note that I'll try to remember to mention again, but just in case:

"Default BT Functions", such as Nickname, NavBar Properties and stuff, are handled 'elsewhere' by BT. You will include them in your config (to be discussed later). Also to be included in your config are the custom properties unique to YOUR plugin. Those will be HTML files, (See #6) and will be kept in the plugin directory root.

Step 1: Make a list of plugin attributes you wish to provide the user.

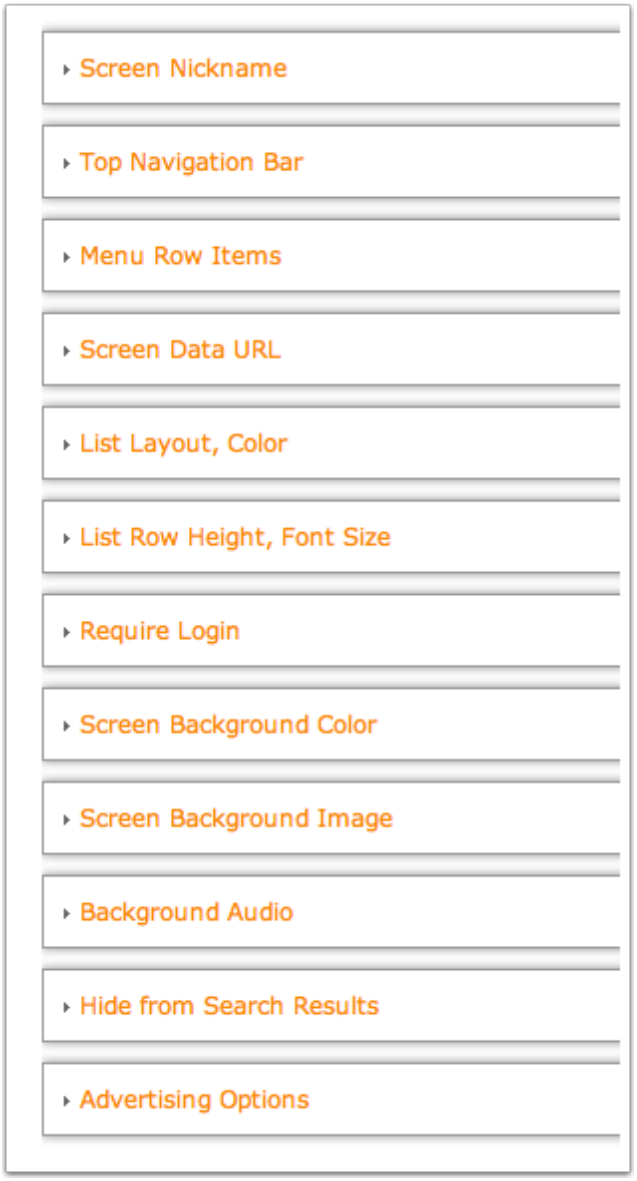


Make a list of attributes you want to provide user control over. This could be images, colors, strings, and almost anything you want. This is implemented using an HTML Form object, typically text boxes or drop down boxes, but it could be anything you want.

Also, get used to the fact that regardless of what you want it to be, while saved in the control panel, and delivered to your app plugin, the data will be A STRING. If you need an integer, a date, or some other kind of data you will need to convert it from String in your app. Not too difficult, usually, but it's something people tend to forget.

"Nickname", "Include in Search", and "backgroundColor" are all different examples of user defined attributes that are usually provided by BT. Anything 'special' will need to be provided by the Plugin Developer.

Quick Question: What's already available?



There are several options already created for you. You can use these as examples, or prepared components in your plugin. Some are going to be necessary, such as 'nickname'. Others, such as 'backgroundAudio' may or may not apply to your particular creation.

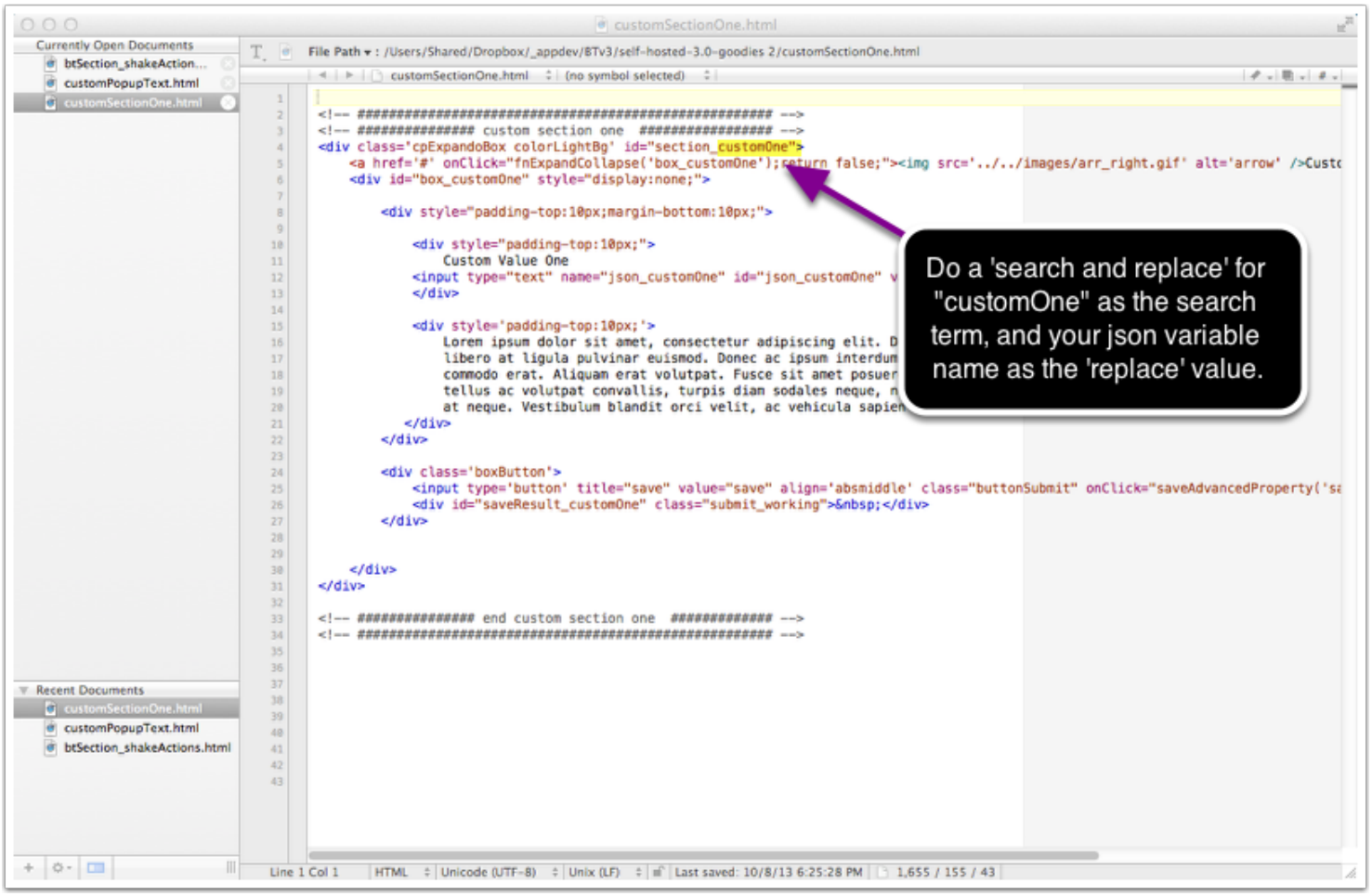
Currently there are 15 'built in' options:

btSection_ads.html - Turns on/off 'iAD' capability (iOS Only)
btSection_backgroundAudio.html - Provides the user a place to select background audio, looping, and basic behavior
btSection_backgroundColor.html - Provides the user ability to choose the screen background color

btSection_backgroundImage.html - Provides the user ability to choose the screen background image
btSection_childItems.html - Provides the user ability to gather additional records of 'child items' for menus, maps, etcetera.
btSection_dataURL.html - Provides the user ability to specify an internet resource for the plugin
btSection_documentBehavior.html - Provides the user ability to specify certain behavior for the document displayed in the plugin
btSection_login.html - Provides the user ability to require users to authenticate, or have already been authenticated, to use this plugin
btSection_menuListLayout.html - Provides the user ability to change the layout appearance of a menu list in iOS (if applicable)
btSection_menuListRows.html - Provides the user ability to change the layout appearance of menu list rows in iOS (if applicable)
btSection_navBar.html - Provides the user ability to change the appearance/behavior of the Navigation Bar
btSection_nickname.html - Provides the user the ability to change the display name of the plugin
btSection_screenJson.html - Provides the user ability to directly edit the plugin Json data
btSection_search.html - Provides the user ability to include/exclude the plugin from being searched
btSection_tabBar.html - Provides the user ability to hide/show the tab bar while plugin is active

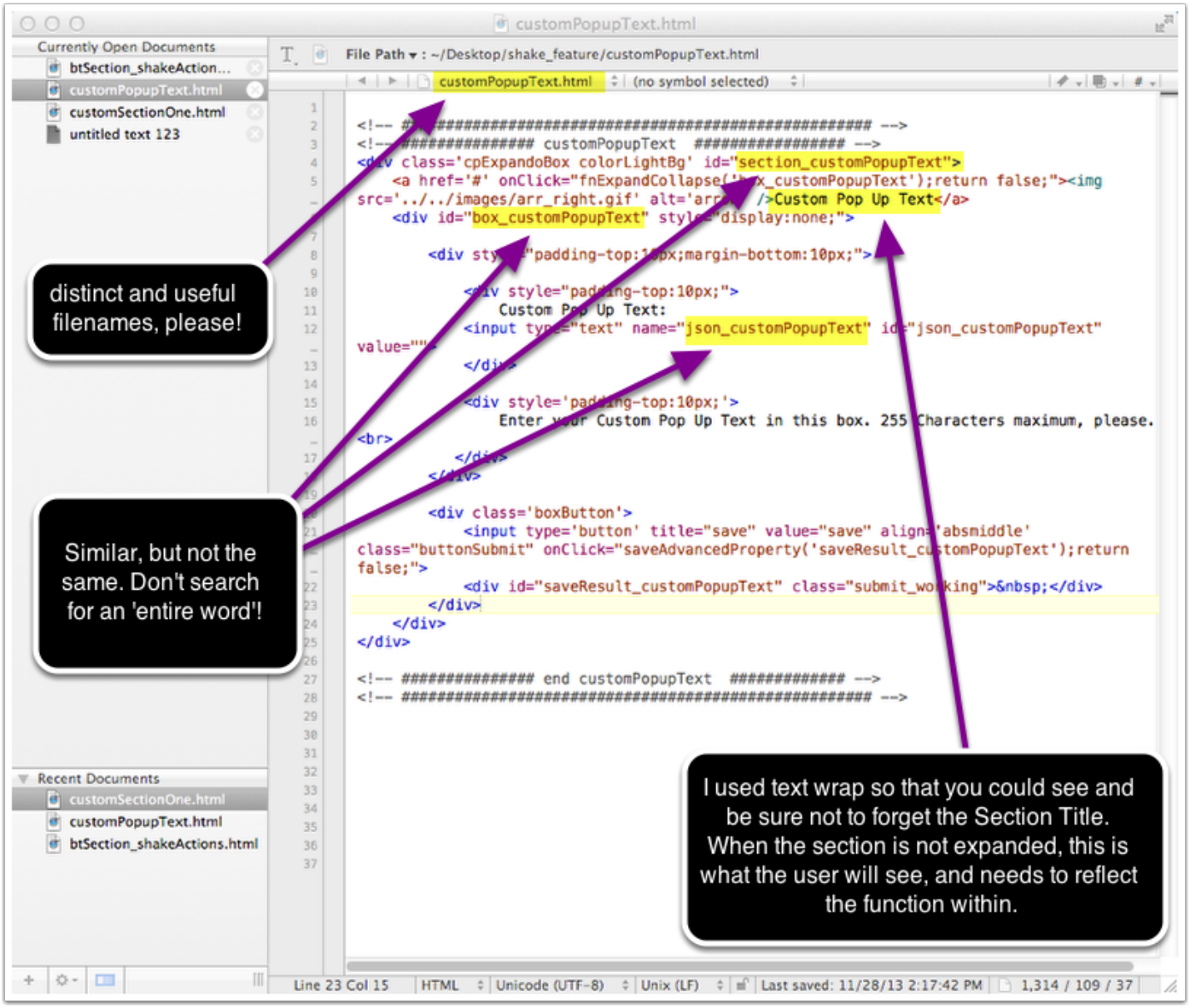
You are probably already familiar with most of these functions in your BT control panel, for various plugins. The ones that are not listed are usually custom made for a particular plugin. In fact, that is why we are here. If any of these provide a 'similar' function that you require, it might be easier to make a copy of an existing "feature", modify it to suit your needs, and save it as a "new" feature for your plugin. If you find yourself in this position, these can be found in your self hosted directory in BT_Root/bt_v15/bt_includes.

Step 2: Start from Scratch



Don't reuse your old PHP driven code; it's probably not going to be of any use here. The surest way is to start from scratch. Get a copy of the sample custom section files. Cant find one? [I placed one in my dropbox just in case.]

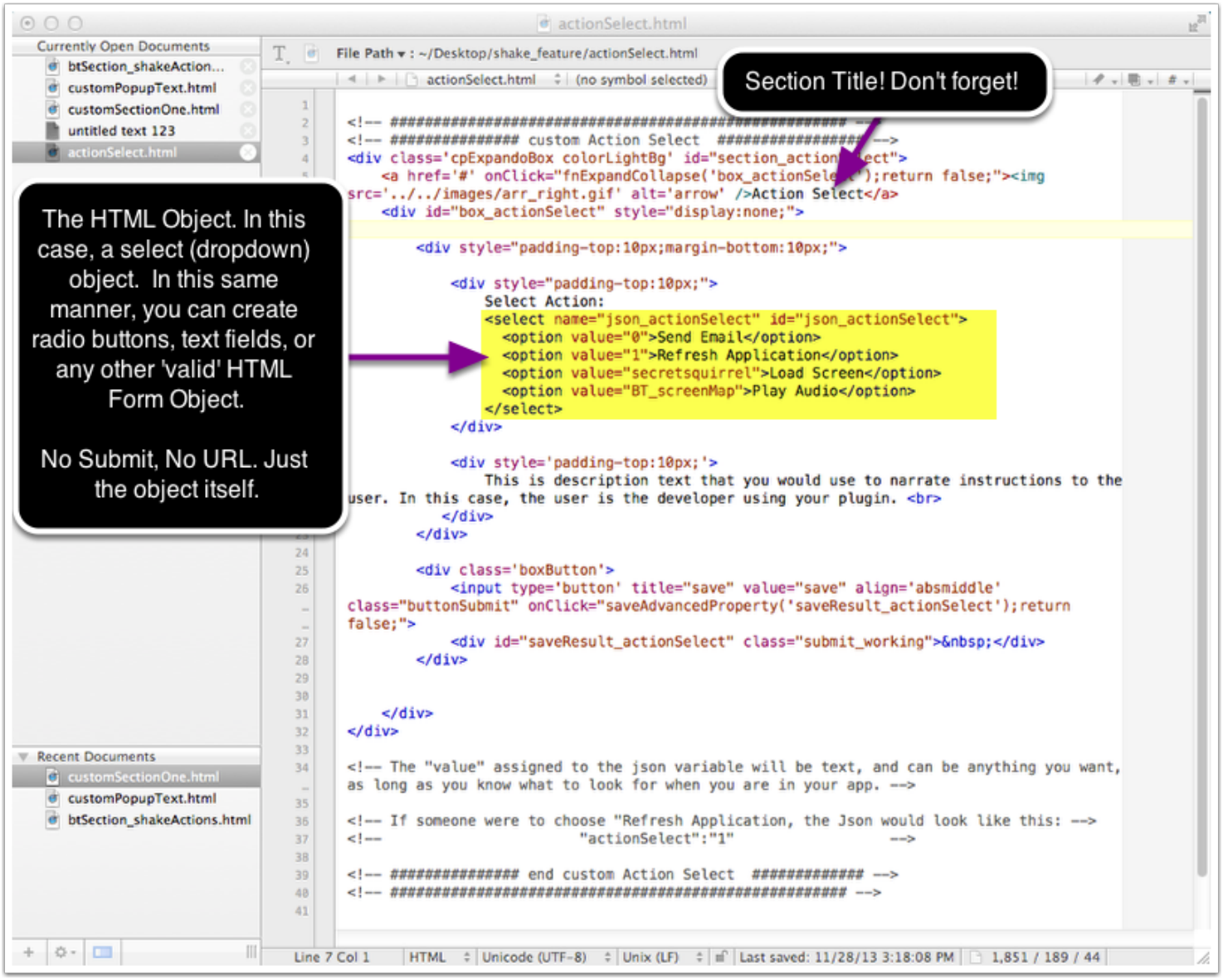
Step 3: Insert your information



Do a search and replace on the text 'customOne' and replace it with your desired variable name. In the above example, we did a search for 'customOne' and replaced it with 'customPopupText'. Do not search for an 'entire word' because the phrase is mixed with other specific parts that we don't want to miss. Replace all occurances of 'customOne'. Typically, if you 'replace all' you should have made 7 changes.

And you're done with this section. Perhaps. For the moment, this example uses a text box. But if you need to use a dropdown box, radio button, or some other type, it's rather easy. Just create it like you would any other HTML Form object. See the next pseudo step.

Step 3A: I need a dropdown box, not a text box



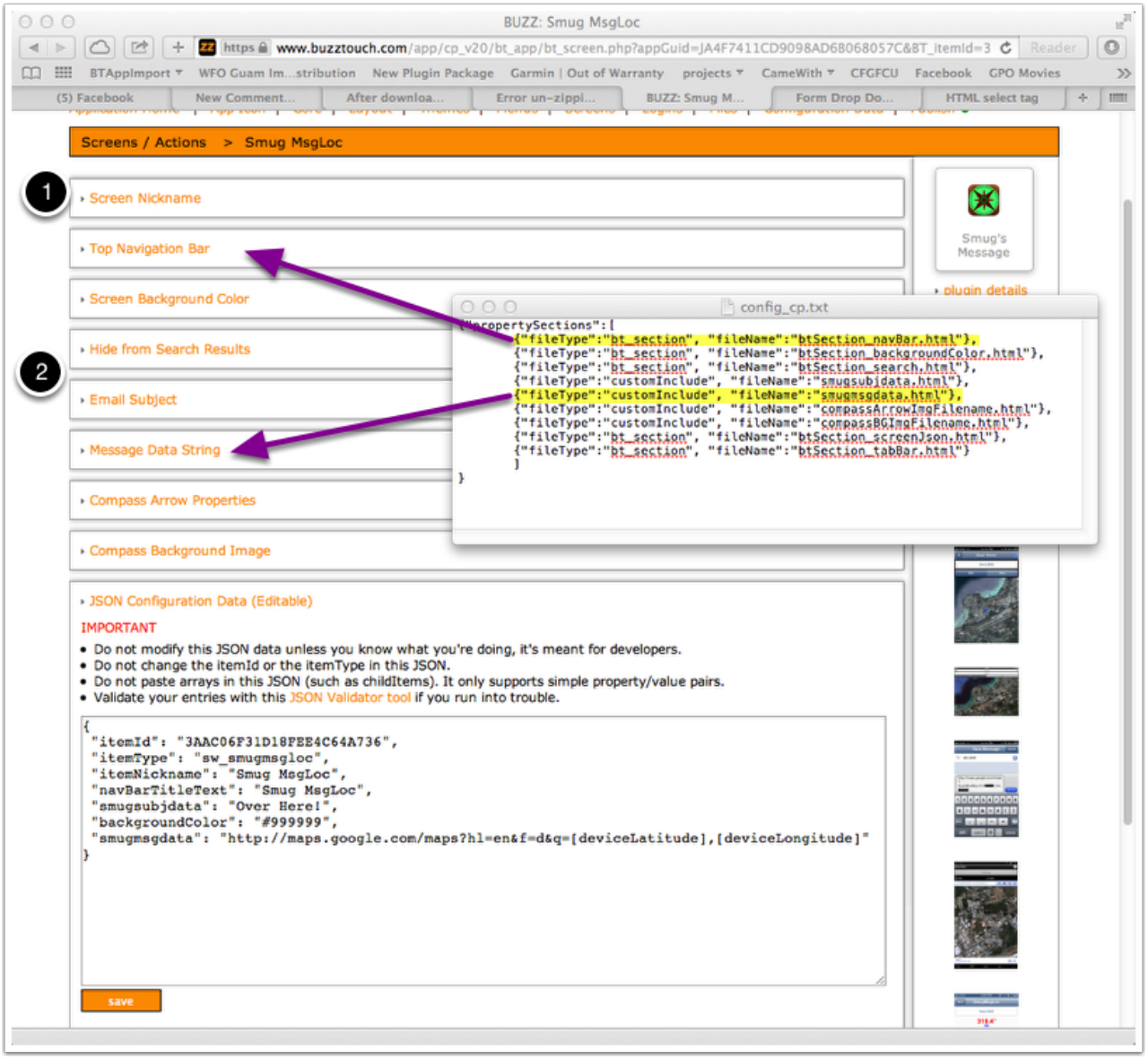
In this example, we're using a drop down box to limit the user to the options we wish to provide them. This can be useful if you want there are only a few specific option choices available.

I chose different values to display a point:
1) It's going to be a string, even if the string is a number.
2) It really doesn't matter what the value is, as long as you know what to expect on the app side. Try to keep it useful though, so it's easy to remember or recognize.

The thing to remember is, it's not rocket science, it's only HTML. BT does the heavy lifting by saving the json values in your database (providing your search/replace was successful).

All you have to do is provide the HTML Form elements to display user options and gather your data.

Step 4: Creating your Control Panel page.



First of all, you no longer 'create' a control panel page. You are creating 'sections' and tying them together in the control panel via a configuration sheet for your plugin. It is called your 'config_cp.txt' file.

1) ALL plugins by default have 'nickname' first. You can't get rid of it, and you don't want to. But you also don't want to repeat it, so leave it out of your config_cp.txt file; it's already provided.
2) The sections are displayed in the order that your config_cp.txt file has them listed. Want to change the order? do it in your config_cp.txt file
3) All of the "BT Provided Options" are located in a different directory than your plugin. When you want to use the BT Function the format is as follows:

{"fileType":"bt_section", "fileName":"<functionName.html>"} - This file is json too, so make sure you add commas if need be. And don't if it's the last entry.

When you want to use one of your custom attributes, you will include the HTML file in your plugin directory, and specify it in the config_cp.txt file as follows:

{"fileType":"customInclude", "fileName":"<customFunctionName.html>"} - same as above; add commas if needed.

Create ONE section/file for each attribute, and list them all in your config_cp.txt file.

Step 5: Getting it into your plugin

```
{
    "itemId": "23690D06BD615F92FAF5F36",
    "itemType": "BT_screen_quiz",
    "itemNickname": "quiz",
    "navBarTitleText": "Pre-Qualifying Quiz!",
    "navBarStyle": "solid",
    "dataURL": "http://apps.marianasgps.com/ev1/duanepahl/prequalquiz.txt",
    "cacheWebImages": "1",
    "quizNumberOfQuestions": "10",
    "quizQuestionDelay": "3",
    "quizRandomizeQuestions": "1",
    "quizShowCorrectAnswers": "1",
    "quizPointsPerAnswer": "10",
    "quizSoundEffectFileNameCorrect": "right.mp3",
    "quizShowTimer": "0",
    "quizFontColorQuestions": "#FFFFFF",
    "backgroundColor": "#13407a",
    "childItems": []
},
```

The image is of a BT_config.txt file, focusing on the Quiz json. From this I can see each 'variable' that was created in the control panel, and the STRING values each has. Each of these variables will be available in the plugin.

Without going too deep into the BT Mentality, let's assume a couple of things:

A) You know how to create a variable in your header file, and synthesize it in your implementation file. If not, learn NOW.
B) Keep in mind all the data you get from your control panel will be a STRING. You're going to need to know how to use that string in your plugin. That could mean various conversion methods, depending if you need to end up with a double, an integer, a URL, an image, or whatever. It 'can' be done. Just look at other plugins that others have created to see how they did it. It's easy 'once you know how'.

Depending on your programming style, you can do one of two things:

A) Create a class variable and assign your value to it most anytime.
B) Create a local variable and use it immediately.

It's your decision. Things like background images, colors and things can be set immediately or later; it really is your choice. But other items you will use as a basis for your plugin behavior, and you may want to create a class variable so you can use it in different methods if you need to.

iOS example of a class defined variable being assigned a value from your json screenData:

```
imageURL = [BT_strings getJsonPropertyValue:self.screenData.jsonVars
nameOfProperty:@"imageURL" defaultValue:@""];
```

Android example of a class defined variable being assigned a value from your json screenData

```
dataURL = BT_strings.getJsonPropertyValue(this.screenData.getJsonObject(), "dataURL", "");
```

an iOS example of a locally defined variable being assigned a value from your json screenData:

```
    NSString *tmpURL = @"";
    tmpURL = [BT_strings getJsonPropertyValue:self.screenData.jsonVars
nameOfProperty:@"dataURL" defaultValue:@""];
```

an Android example of a locally defined variable being assigned a value from your json screenData:

```
String dataURL = BT_strings.getJsonPropertyValue(this.screenData.getJsonObject(), "dataURL", "");
```

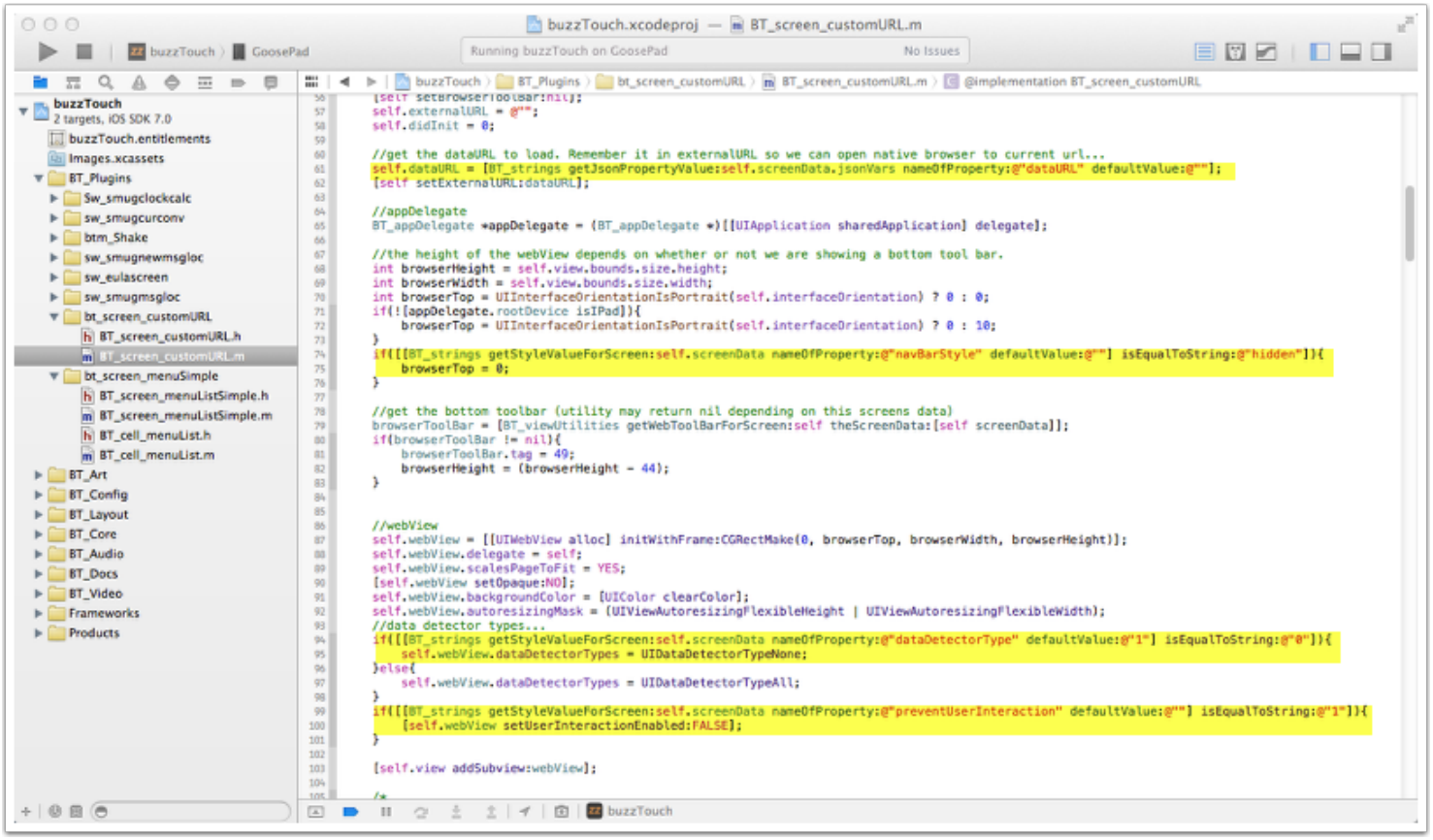
an iOS example of no variable defined; just immediate use of a json variable in code:

```
    if([[BT_strings getJsonPropertyValue:self.screenData.jsonVars nameOfProperty:@"includeAds"
defaultValue:@"0"] isEqualToString:@"1"]){
        [self createAdBannerView];
    }
```

You'll notice in the above examples that the only real difference between the local variable and the class variable is that class variable values retain throughout the class, and a local variable retains value only in the method from which it is declared. Local variables are typically more efficient that global variables, from a memory management sense.

If you've never done it before, it seems daunting. But it's not as bad as you might think.

Screenshot example...



Various examples of the use of control panel options in a plugin...

A Note on Child Items

To be honest, I haven't messed too much with child items, and so for that particular skill set, I have nothing to say. I will amend this tutorial when I get up to speed on it, or encourage others to write one at any time!

Cheers!

-- Smug